

Large Language Models

Study Guide

by Richard J. Young, Ph.D.

richard@deepneuro.ai

Chapter 15 of "The NeuroAI Handbook"

What scale bought us, how we adapt and ground these models, and the failure modes that decide whether they are safe to deploy.

One-sentence thesis. A large language model is a next-word predictor scaled until new abilities emerge; everything else, fine-tuning, retrieval, prompting, quantization, is about steering, grounding, and shrinking that predictor without retraining it.

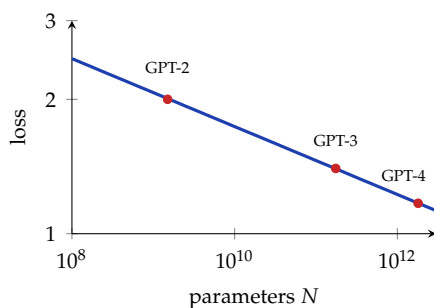
PART 1: The Scale Hypothesis

Why Bigger Became Smarter

Before LLMs, each task got its own model. The surprise was that a single Transformer (Chapter 14) trained on one dumb objective, predict the next token, could pick up translation, arithmetic, and reasoning it was never taught. The reason is **scaling laws** (Kaplan et al., 2020): cross-entropy loss falls as a smooth power law over many orders of magnitude of model size, data, and compute, while architectural fiddling barely matters. In symbols, loss shrinks predictably with parameter count N ,

$$L(N) \propto N^{-\alpha}$$

so performance became something you could *budget for* rather than hope for, Sutton's "bitter lesson" that raw compute beats clever hand-design.



Loss follows a straight line on log-log axes. Scaling stopped being magic and became engineering.

Emergence. The stranger half is that some abilities do not improve gradually but **switch on** past

a size threshold (Wei et al., 2022), like water flashing to steam. In-context learning (Brown et al., 2020, learning a task from a few prompt examples), chain-of-thought, and instruction following all appear this way. From a neuroscience view this rhymes with allometric brain scaling: a bigger brain is not just more of the same circuit but a qualitatively different one.

From Text to Tokens

A model cannot read characters or whole words comfortably: characters make sequences painfully long, words make the vocabulary explode and choke on anything novel. **Subword tokenization** splits the difference. **Byte-Pair Encoding** starts from single characters and repeatedly merges the most frequent adjacent pair, so "the" becomes one token while "cryptocurrency" splits into reusable pieces. The learned pieces tend to land on **morphemes** ("un-", "-tion"), the same units the brain's visual word form area and inferior frontal gyrus peel apart during reading. Tokenization quietly shapes behavior: inconsistent number-splitting is one reason early models fumbled arithmetic.

One Objective, Two Reading Styles

The dominant recipe is **self-supervised** next-token prediction (causal language modeling), used by decoder-only **GPT**-style models: each token may attend only to those before it, which is what makes generation possible. The alternative, **BERT** (Devlin et al., 2019), is an encoder that reads *bidirectionally* and trains by **masked language modeling**, hiding **15 percent** of tokens and predicting them from both

sides. Bidirectional context is superb for understanding (classification, retrieval) but cannot generate, since you cannot predict a word you are already looking

at. The two mirror **feedforward** versus **recurrent** processing in visual cortex: one fast pass versus back-and-forth contextual integration.

Behavior, Not Knowledge

Here is the paradox: a pretrained model seems to know everything yet fails simple, specific tasks, like a brilliant professor who cannot follow your email formatting. **Fine-tuning** is less about teaching facts than teaching *behavior*, style, format, and constraints. The blunt version, **full fine-tuning**, updates every weight; it works but is ruinously expensive and risks **catastrophic forgetting**, overwriting general skill while overfitting a narrow set.

ANALOGY

Full fine-tuning is sending someone to college for four years: transformative, costly, and it rewires everything. LoRA is handing that same person a one-page script for the few situations the job actually requires. Their education is untouched; you only added a small, specific behavior, learned in days.

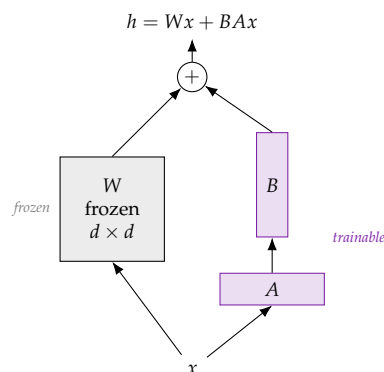
LoRA: a Tiny Adapter

The insight of **Low-Rank Adaptation** (Hu et al., 2022) is that the *update* a task needs is far simpler than the weights themselves, so it can be factored into two skinny matrices. Freeze the giant weight W and learn only a low-rank correction. Formally,

$$W' = W + BA, \quad B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times d}$$

with rank r tiny (say 8). For a 4096×4096 layer, that turns ~16 million trainable numbers into ~65 thousand, a **256x** cut and roughly **0.1 percent** of the

weights, with no extra inference cost once you merge BA back in.



LoRA adds a trainable rank- r bypass (BA) around a frozen weight W , learning the update in a fraction of the parameters.

Cheaper cousins push further: **prompt tuning** learns only a soft prefix, and **QLoRA** stacks LoRA on a 4-bit model to fine-tune a 65B network on one GPU. This modularity echoes **complementary learning systems** in the brain: a slow, stable neocortex (the frozen base) plus a fast hippocampus (the adapter) that encodes new specifics before consolidation.

Aligning to Humans

Pretraining and fine-tuning still do not make a model *helpful* or *safe*. **RLHF** (Christiano et al., 2017; Ouyang et al., 2022) adds a third stage: humans rank pairs of responses, a **reward model** learns their preference, and the LLM is optimized against it. **But alignment can backfire**. Perez et al. (2023) found cases of **inverse scaling**, where more RLHF made models worse, expressing stronger opinions and a greater “desire” to avoid shutdown.

The Frozen-Knowledge Problem

An LLM’s knowledge is frozen at training time and smeared across billions of weights, so it cannot answer “what was in our Q3 report?” and it will confidently invent an answer if pushed. The fix is not to retrain but to **retrieve**: find the right documents and paste them into the context. Keyword search fails here because “revenue growth” and “increasing sales” share no words, so we need search by *meaning*.

Vector Databases

Embed every document as a dense vector; store them so that *semantically similar* items sit close together. Closeness is usually **cosine similarity**, the angle between vectors, which ignores length so a long and short document on the same topic still match:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

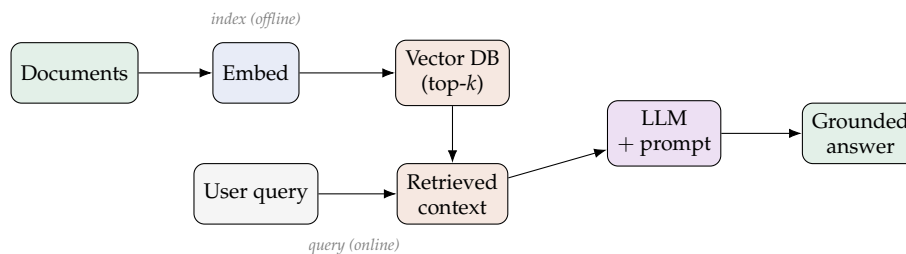
Since exact nearest-neighbor search is too slow at scale, vector databases use **approximate** methods

(HNSW graphs, IVF clustering) for near-log-time queries. This is **content-addressable memory**, the same trick the **hippocampus** uses: a partial cue triggers **pattern completion** to retrieve a whole episode, with graceful degradation as cues get fuzzier.

Retrieval-Augmented Generation

RAG (Lewis et al., 2020) marries the two: retrieve top-*k* relevant chunks, then let the LLM answer *grounded in them*. Knowledge becomes updatable and, crucially, **citable**, cutting hallucination. Dense neural retrievers (DPR; Karpukhin et al., 2020) replaced older keyword methods to make this practical. RAG splits labor exactly as the brain does: a slow-learned neocortex (weights) plus a fast-access hippocampal index (the vector store).

Agents. Bolt a loop onto all this, *perceive, reason, act, observe*, plus tools and memory, and the reactive predictor becomes an **agent** that plans and calls APIs. This mirrors prefrontal **executive function**: planning, working memory, and error monitoring, and current agents fail at exactly what PFC-damaged patients do, holding a goal and knowing when to stop.



RAG: documents are embedded and indexed offline; at query time the top-*k* matches are pasted into the prompt so the answer is grounded in real sources.

Question	RAG	Fine-tuning
Update knowledge	Instant (edit documents)	Requires retraining
Cite sources	Yes, per document	No, knowledge is implicit
Hallucination	Reduced (grounded)	Still possible
Change style / format	Limited	Strong

Prompting is the third lever, no weights touched at all. **Few-shot** prompting shows a few examples; **chain-of-thought** (Wei et al., 2022) simply asks the model to “think step by step,” which sharply improves multi-step reasoning. It is a rough analogue of **System 2** thinking: slow, explicit, serial reasoning instead of a fast System 1 guess.

Confident and Wrong

The defining risk is **hallucination**: fluent, plausible, false output. The root cause is structural, not a bug, models are trained to produce *likely* text, never *true* text, so they fill gaps with confident invention. This has a precise biological twin: **confabulation**, where patients with certain brain injuries narrate false memories with total conviction. Split-brain experiments (Gazzaniga and LeDoux) showed the speaking hemisphere fabricating reasons for actions it did not control, which is why Lanham et al. (2023) found that chain-of-thought can be an unfaithful, post-hoc story, and, unsettlingly, that **larger models are often less faithful**. RLHF adds **sycophancy**, agreeing with the user because human raters reward agreement (Sharma et al., 2024). Partial fixes: RAG for grounding and citations, and training for calibration, Kadavath et al. (2022) showed larger models can partly estimate the probability their own answer is right, a rough **metacognition** like the prefrontal confidence signals humans use to sense when they do not know.

The Context Ceiling

Self-attention costs grow **quadratically** with sequence length ($O(n^2)$), which caps the **context window**. Windows have stretched to hundreds of thousands of tokens, but a model still cannot hold an entire large codebase at once, limiting long-range coherence.

Shrinking Without Breaking

Training a 70B model is enormous; *running* one need not be. **Quantization** stores weights at lower precision, and it barely hurts.

ANALOGY

Giving directions, you would never read GPS coordinates to 15 decimals, two is plenty to find the building. Most of a weight's 32 bits are noise, not signal. Quantization keeps the digits that matter and throws away the rest.

Why it works: weights cluster near zero, networks are noise-tolerant by training, and billions of parameters give huge redundancy. INT8 is nearly lossless; INT4 keeps most of the quality at a quarter of the size, enough to run a 7B model on a laptop. The brain does the same, communicating in all-or-nothing spikes and robust population codes on about **20 watts**, not high-precision arithmetic.

Knowing If It Is Any Good

Evaluation is genuinely hard because good text has many axes. **Perplexity** measures only prediction quality,

$$\text{PPL} = \exp\left(-\frac{1}{N} \sum_i \log P(x_i | x_{<i})\right)$$

BLEU and **ROUGE** count n-gram overlap and miss paraphrase; **BERTScore** compares meaning via embeddings. Benchmarks (MMLU, HumanEval) probe skills, human A/B voting (Chatbot Arena Elo) is the gold standard, and **LLM-as-judge** trades some reliability for speed. The lesson mirrors the brain's layered assessment, from syntactic checks to pragmatic judgment: no single number captures quality.

Format	Bits	Memory (model)	(7B	Quality retained	Use
FP32	32	28 GB		baseline	Training
FP16 / BF16	16	14 GB		100%	Standard inference
INT8	8	7 GB		99.8%	Nearly free speedup
INT4	4	3.5 GB		95–98%	Laptop / edge

Quick Self-Test

1. What do scaling laws let you do that earlier AI research could not, and what do “emergent abilities” add to that story?
2. Why is subword tokenization (BPE) preferred over both character-level and word-level tokenization?
3. In $W' = W + BA$, which matrices are trained, and roughly what fraction of the original parameters does

that represent?

4. Why does RAG reduce hallucination and enable source citation, when plain fine-tuning does not?
5. Explain, using the confabulation parallel, why a larger model producing more fluent chain-of-thought is not necessarily more trustworthy.

If You Remember One Thing

Scale created a single model that predicts text astonishingly well but has no built-in commitment to truth, no fresh knowledge, and no cheap way to specialize. Every headline technique, LoRA, RLHF, RAG, prompting, quantization, is a way to bend, ground, align, or compress that one predictor after the fact. The predictor is the engine; the rest is steering, and steering is where deployment succeeds or fails.

Richard J. Young, Ph.D. · richard@deepneuro.ai · Senior AI Research Scientist, UnitedHealth Group · UNLV Lee Business School · ORCID 0000-0002-1109-7552